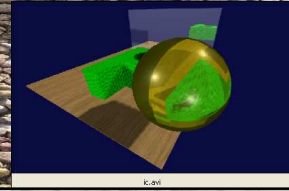


Mike Bailey
Oregon State University
mjb@cs.oregonstate.edu

Steve Cunningham
Brown/Cunningham Associates
rsc@cs.csustan.edu

Introduction to Computer Graphics



Mike Bailey



- Professor of Computer Science, Oregon State University
- Has worked at Sandia Labs, Purdue University, Megatek, San Diego Supercomputer Center (UC San Diego), and OSU
- Has taught over 4,000 students in his classes
- mjb@cs.oregonstate.edu



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Steve Cunningham



- Retired Professor of Computer Science, California State University Stanislaus
- Has served as chair of both the SIGGRAPH Education Board and the Eurographics Education Board
- Has written 7 books on computer graphics topics
- rsc@cs.csustan.edu



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Course Goals



- Provide a background for papers, panels, and other courses
- Create a common understanding of computer graphics vocabulary
- Help appreciate the images you will see
- Get more from the Exhibition
- Provide pointers for further study



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Specific Topics



- The Graphics Process
- Graphics Hardware
- Modeling
- Rendering
- GPU Shaders
- Finding More Information



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Schedule



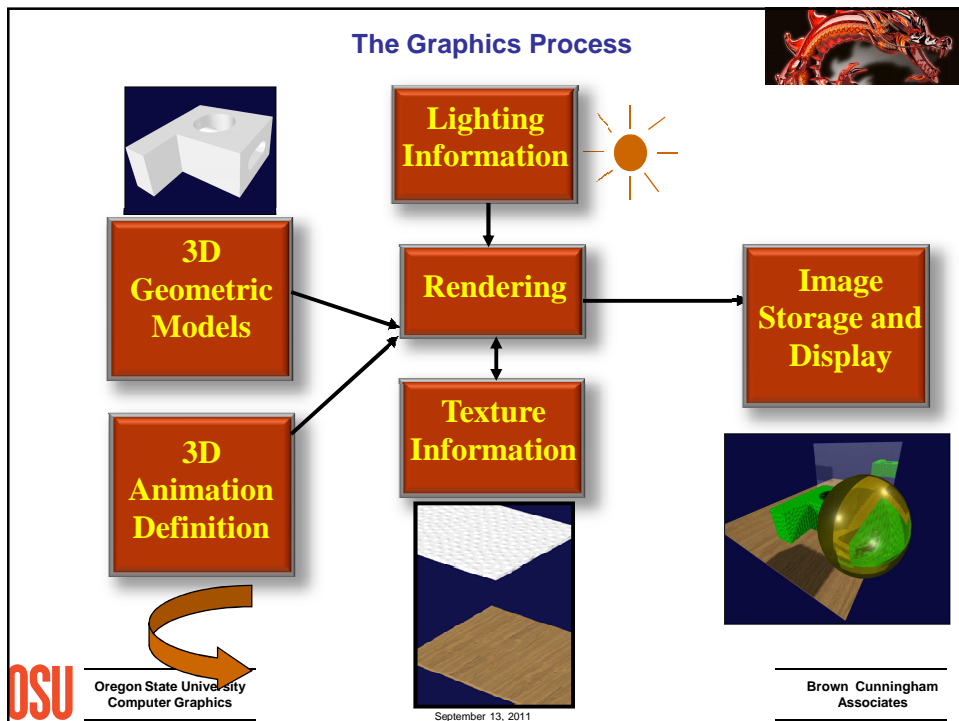
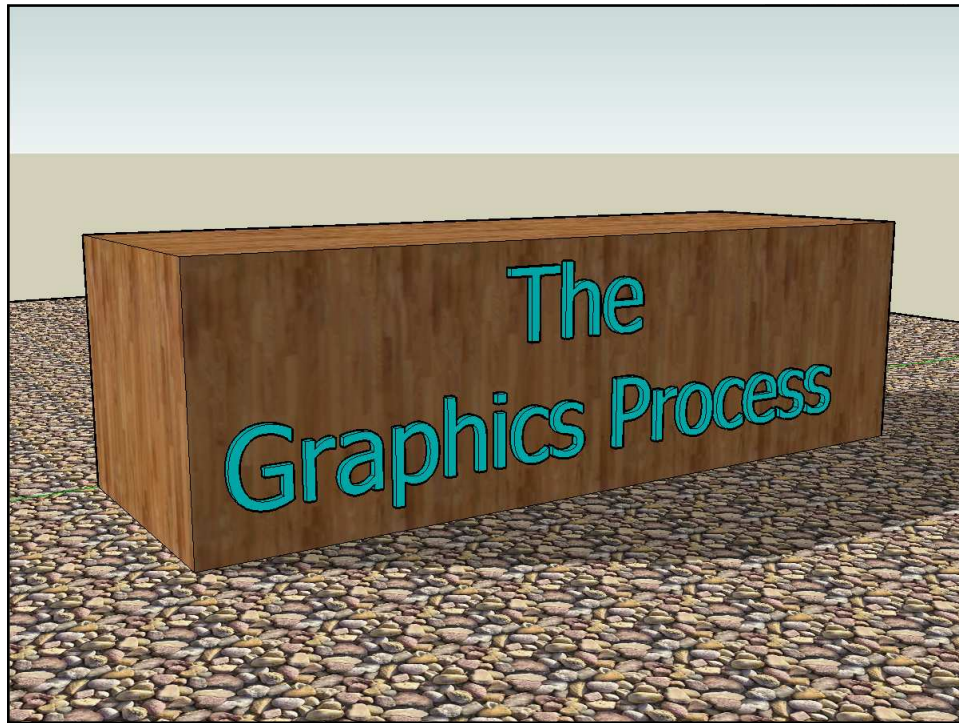
- 0:00 Welcome and Overview
- 0:10 The Graphics Process
- 0:40 Graphics Hardware
- 1:10 Modeling
- 1:45 Break
- 2:00 Maybe our vision isn't as good as we think it is ☺
- 2:15 Rendering
- 2:30 GPU Shaders
- 2:50 Finding Additional Information
- 3:10 Discussion and Questions
- 3:30 Finish

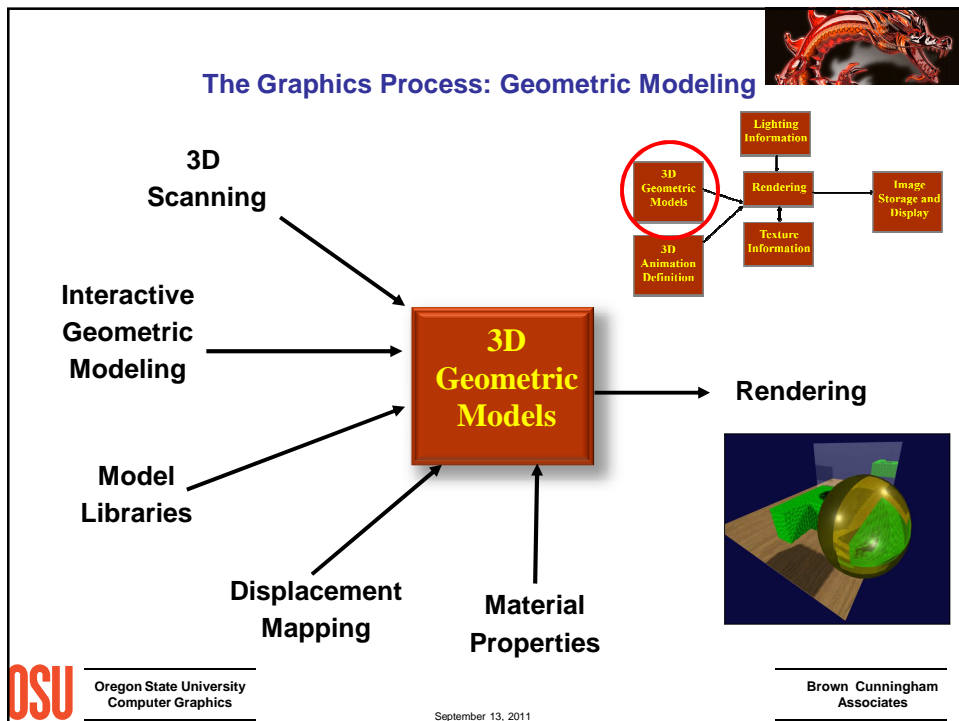
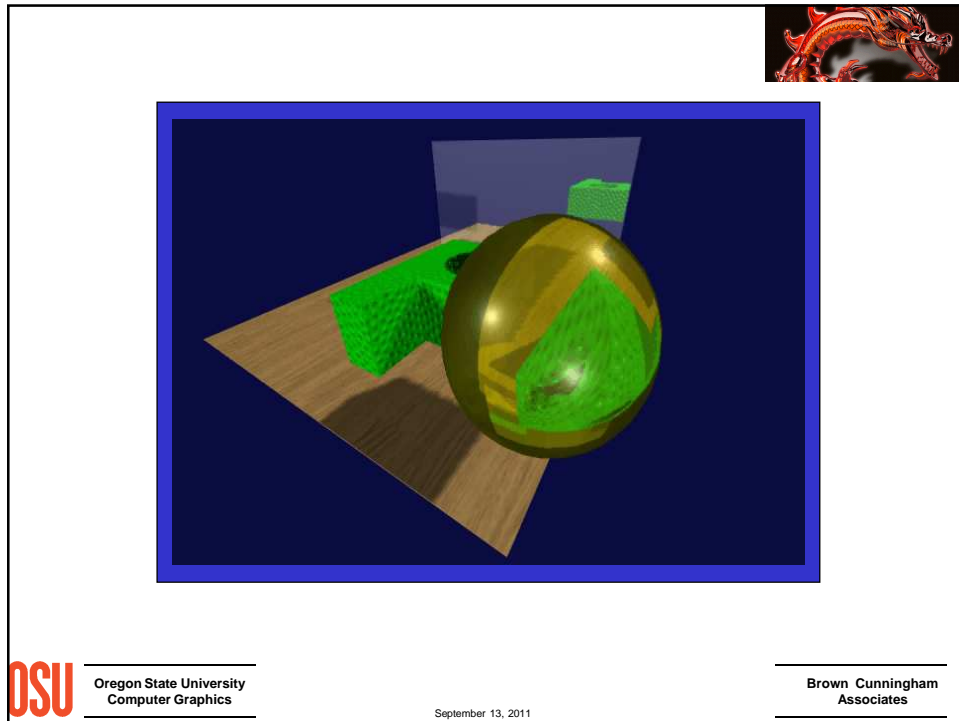


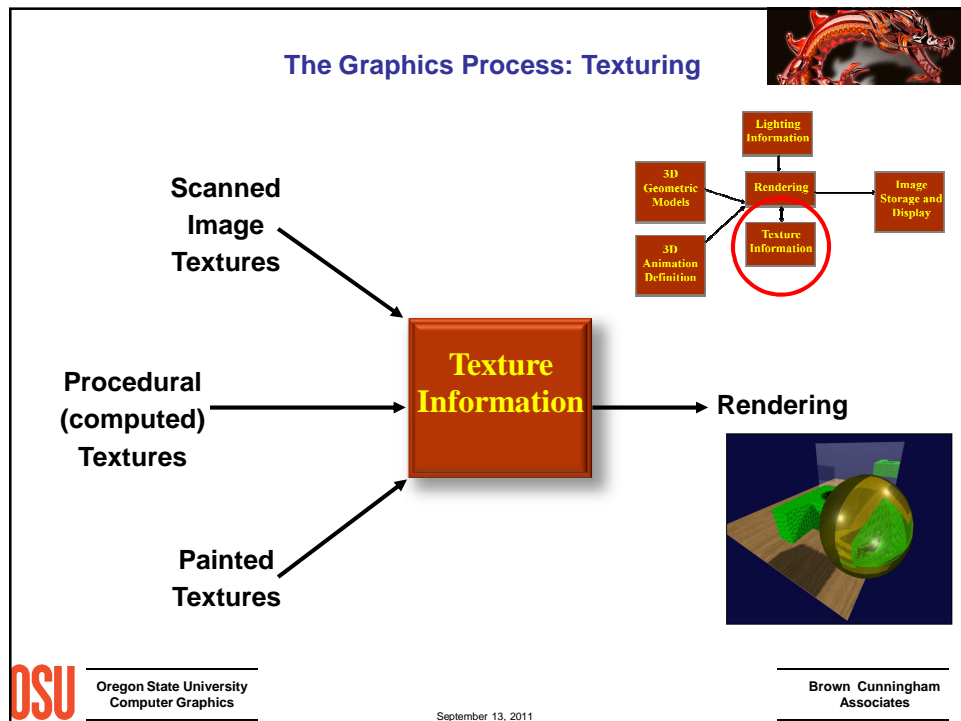
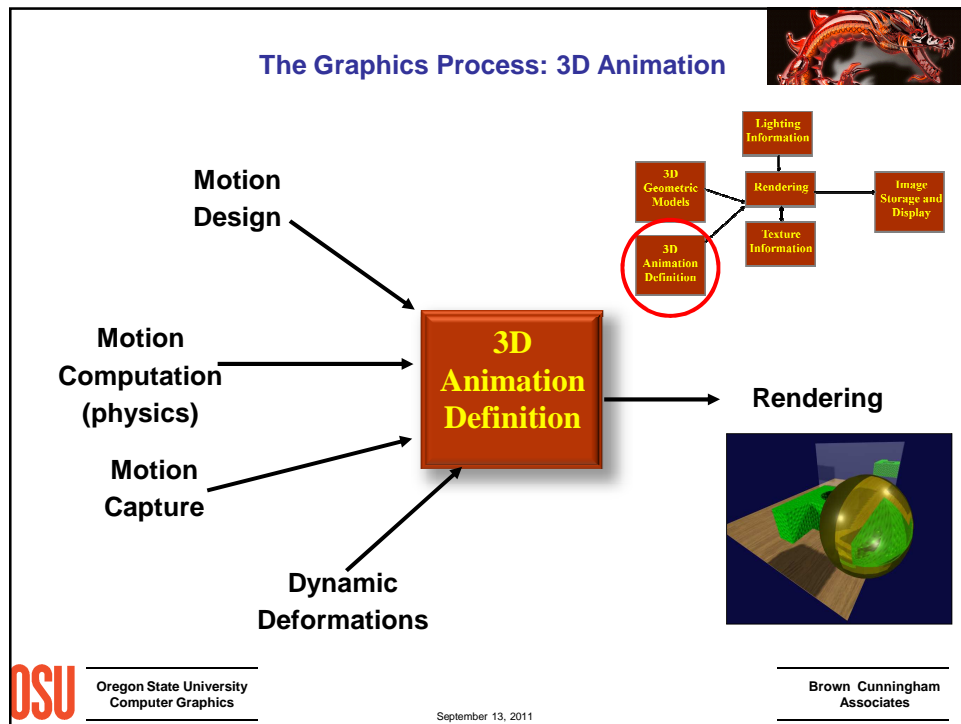
Oregon State University
Computer Graphics

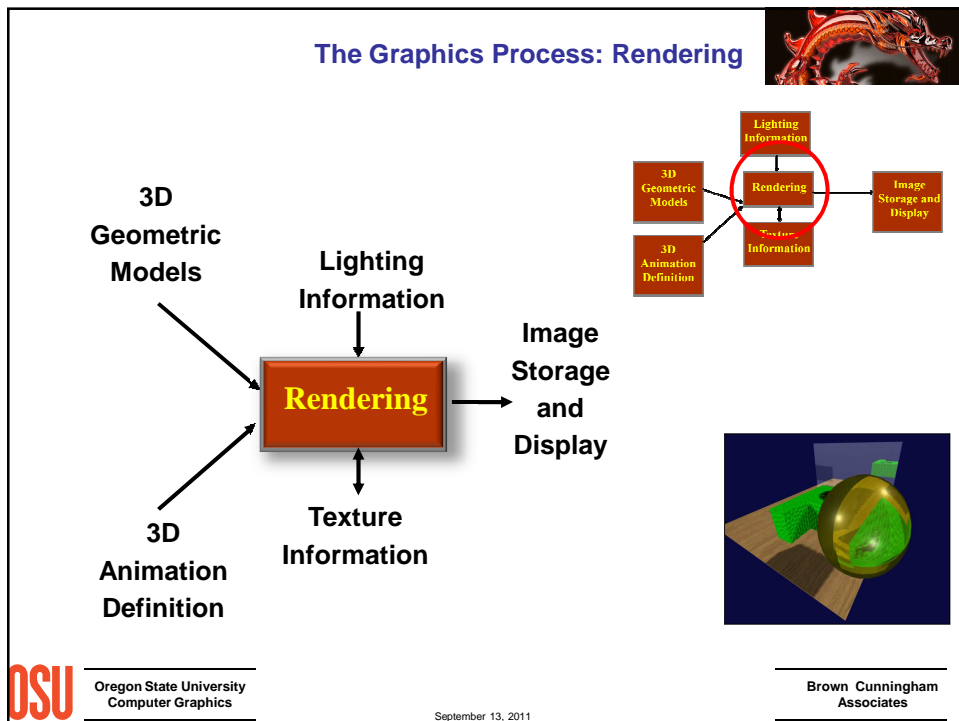
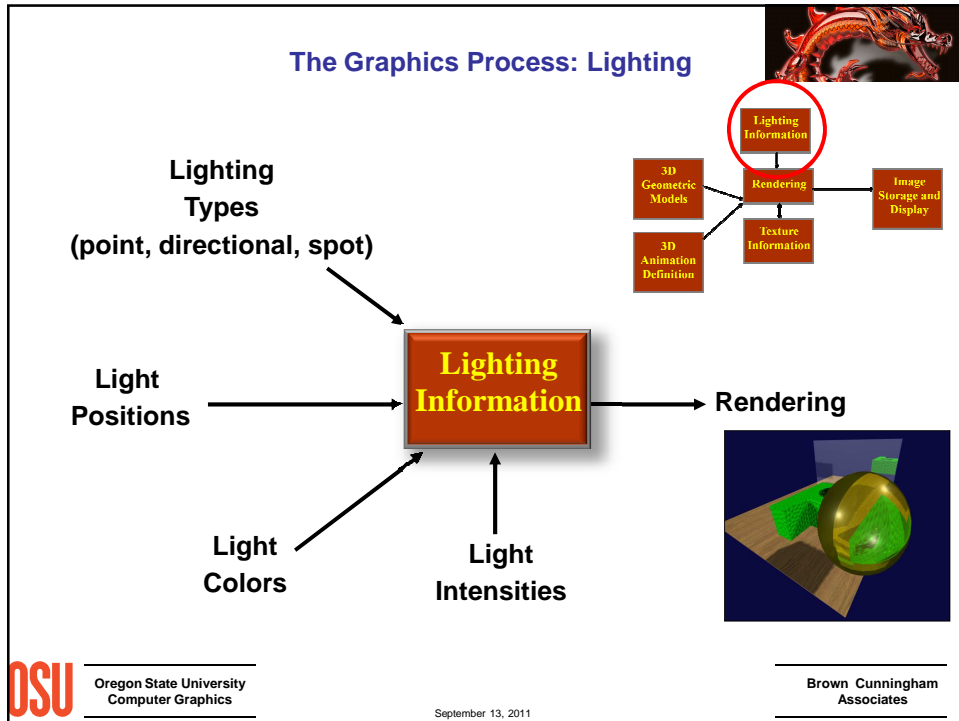
September 13, 2011

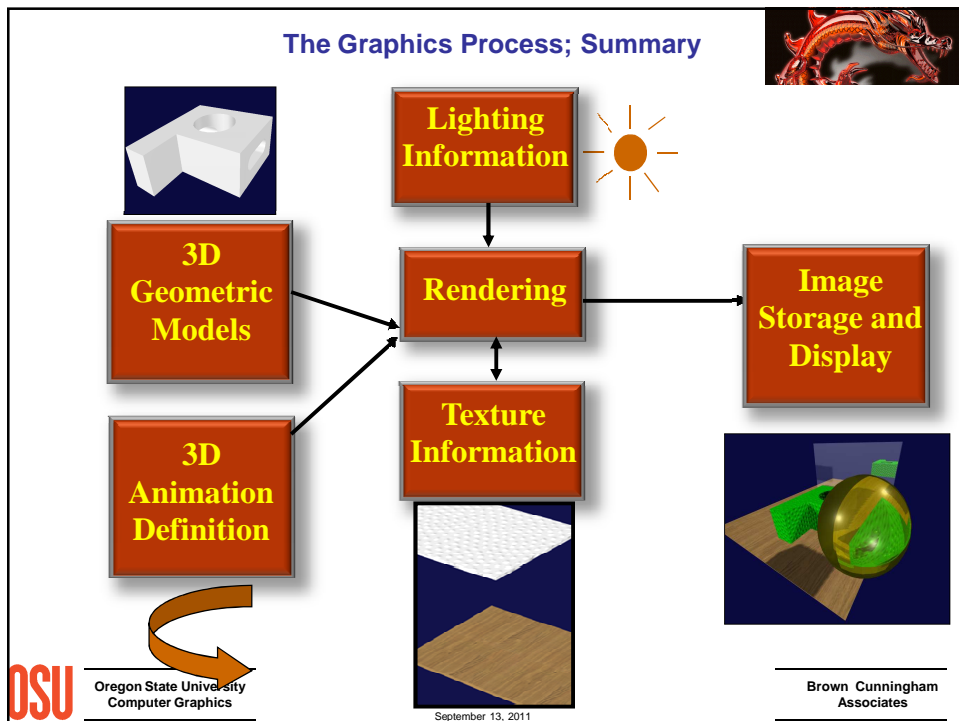
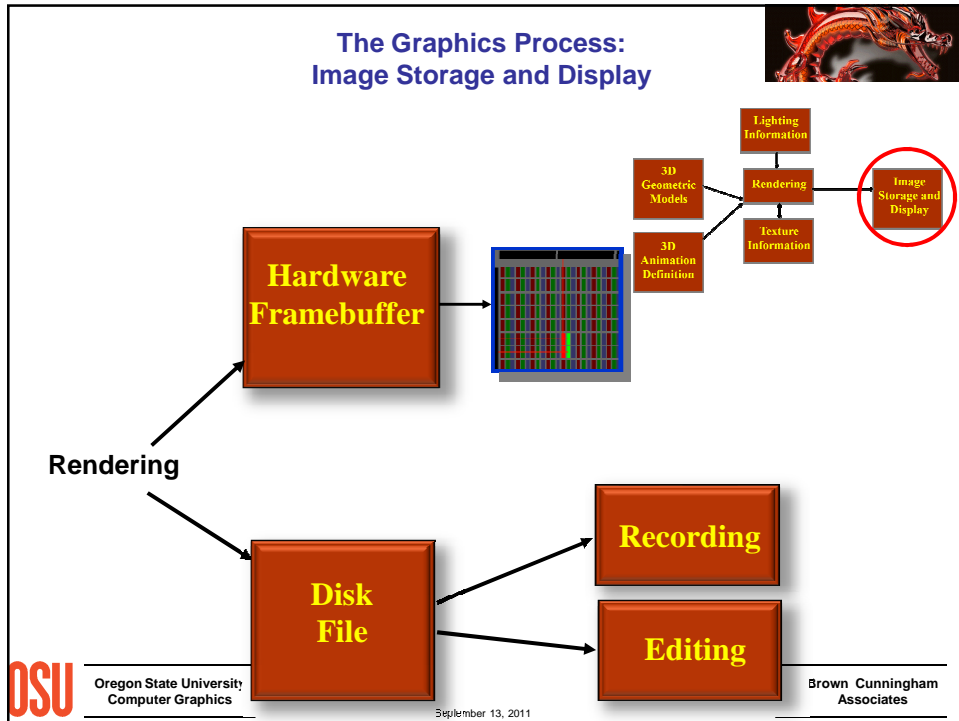
Brown Cunningham
Associates

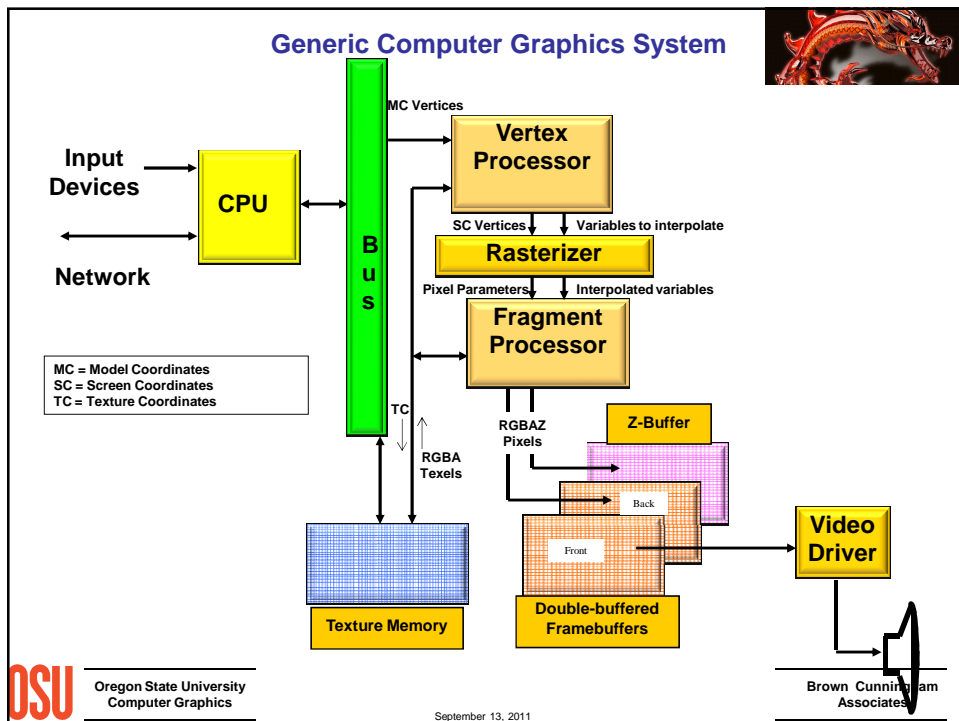
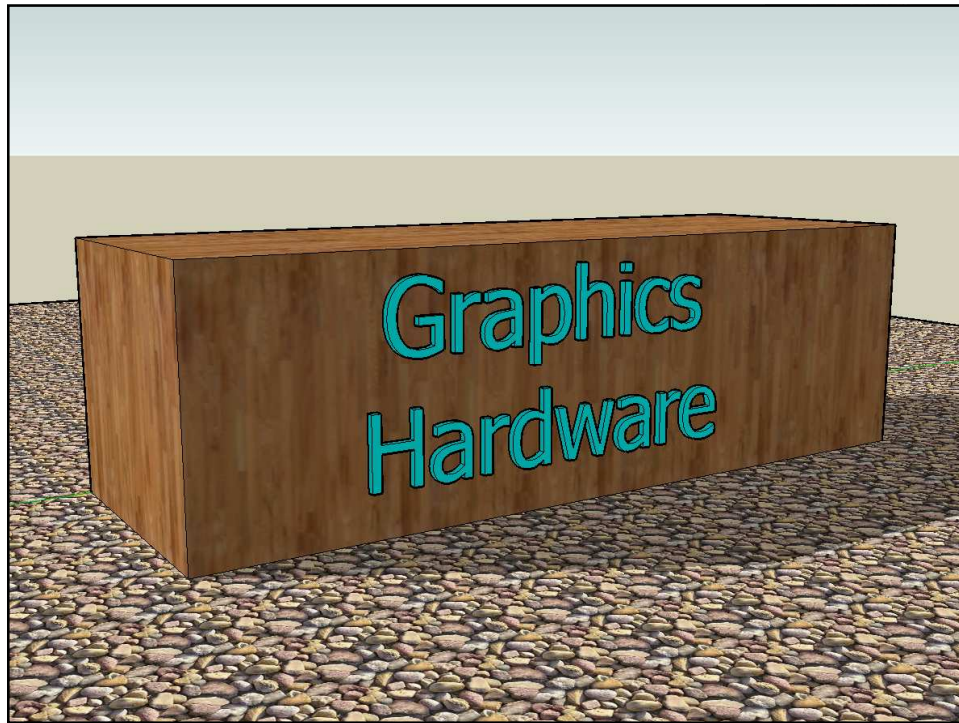




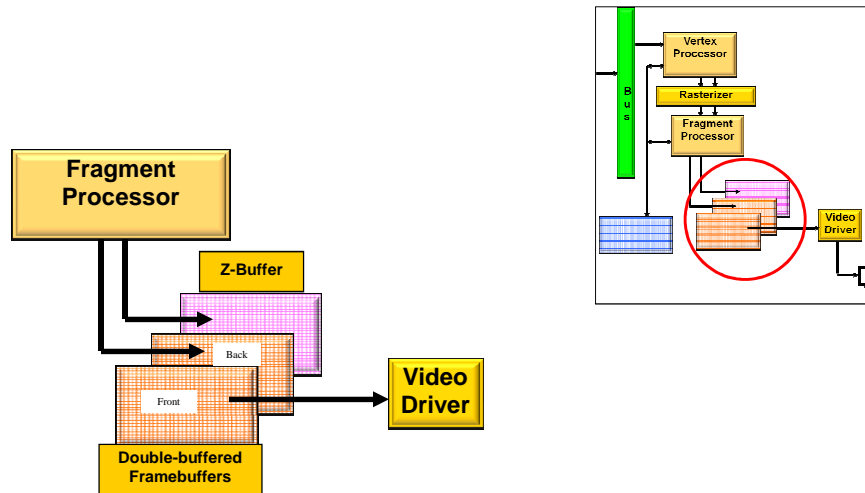








The Framebuffer



Oregon State University
Computer Graphics

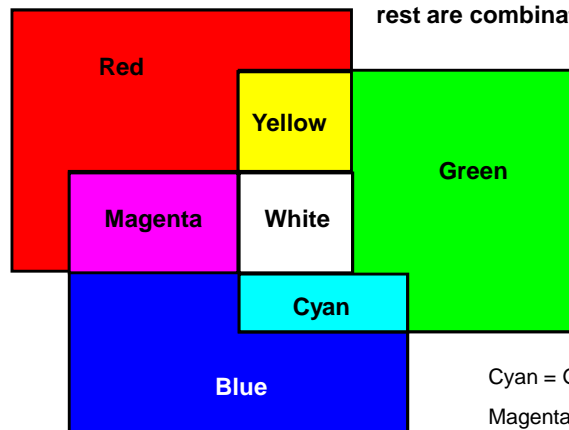
September 13, 2011

Brown Cunningham
Associates

The Framebuffer Uses Additive Colors (RGB)



Red, Green, and Blue are provided. The rest are combinations of those three.



Cyan = Green + Blue

Magenta = Red + Blue

Yellow = Red + Green

White = Red + Green + Blue

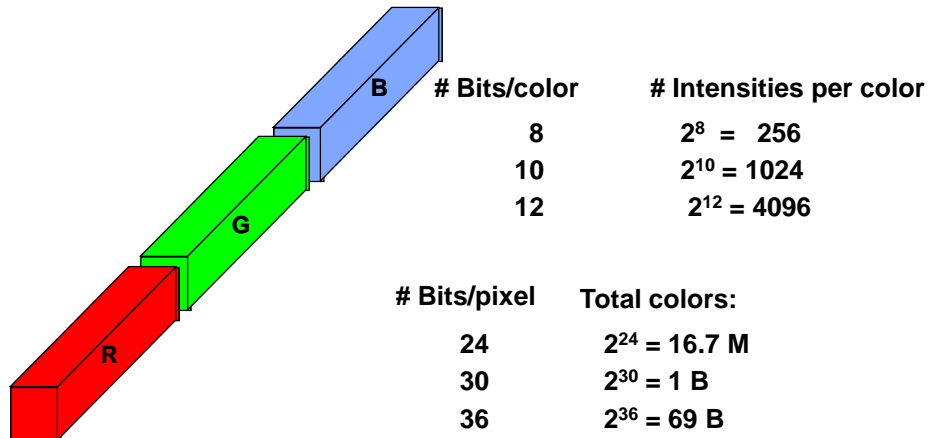


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

The Framebuffer: Integer Color Storage



Oregon State University
Computer Graphics

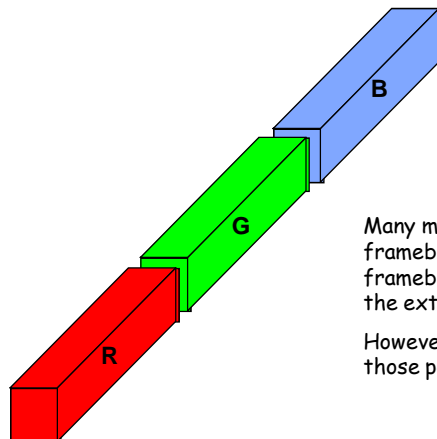
September 13, 2011

Brown Cunningham
Associates

The Framebuffer: Floating Point Color Storage



- 16- or 32-bit floating point for each color component



Why so much?

Many modern algorithms do arithmetic on the framebuffer color components, or treat the framebuffer color components as data. They need the extra precision during the arithmetic.

However, the display system cannot display all of those possible colors.



Oregon State University
Computer Graphics

September 13, 2011

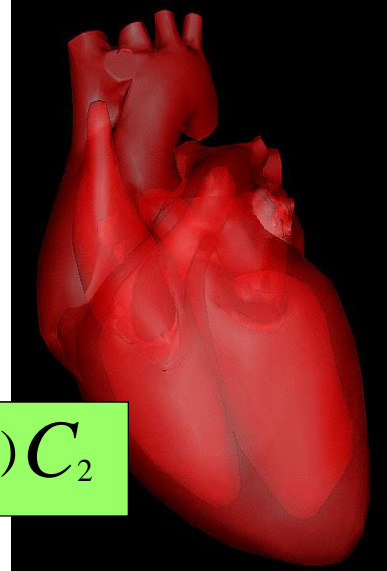
Brown Cunningham
Associates

The Framebuffer



- **Alpha** values
 - Transparency per pixel
 $\alpha = 0$. is invisible
 $\alpha = 1$. is opaque
 - Represented in 8-32 bits
(integer or floating point)
 - Alpha blending equation:

$$Color = \alpha C_1 + (1 - \alpha) C_2$$



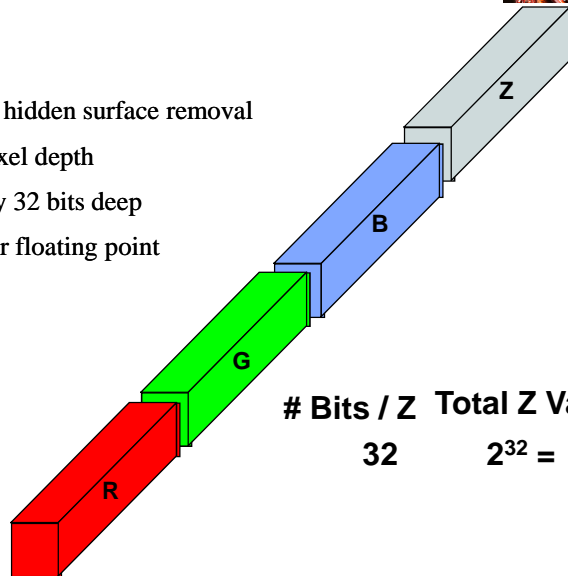
0.0 $\leq \alpha \leq$ 1.0
Oregon State University
Computer Graphics

September 13, 2011

The Framebuffer



- **Z-buffer**
 - Used for hidden surface removal
 - Holds pixel depth
 - Typically 32 bits deep
 - Integer or floating point



Bits / Z Total Z Values:
32 $2^{32} = 4 \text{ B}$



Oregon State University
Computer Graphics

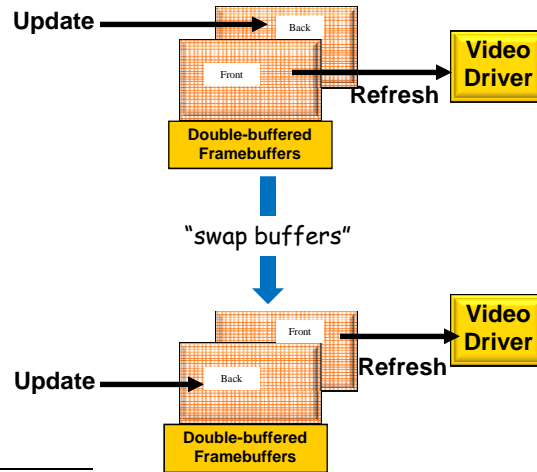
September 13, 2011

Brown Cunningham
Associates

The Framebuffer



Double-buffering: Don't let the viewer see any of the scene until the entire scene is drawn

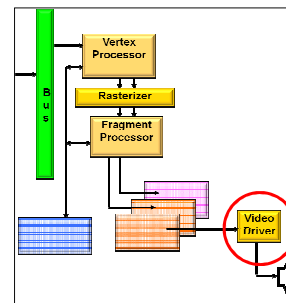
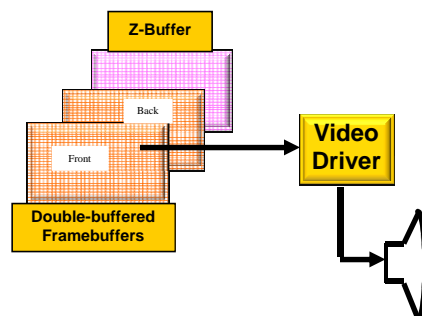


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

The Video Driver



Oregon State University
Computer Graphics

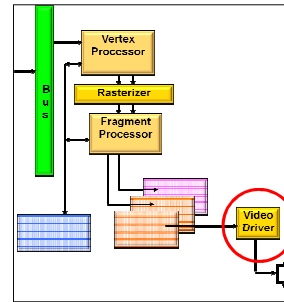
September 13, 2011

Brown Cunningham
Associates

The Video Driver



- N **refreshes/second** (N is usually between 50 and 100)
- Framebuffer contains the R,G,B that define the color at each pixel
- Cursor
 - Appearance is stored near the video driver in a “mini-framebuffer”
 - x,y is given by the CPU
- Video input

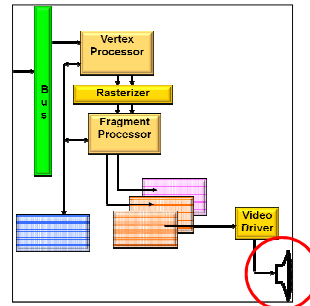
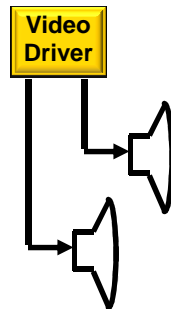


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

The Computer Graphics Monitor(s)



Oregon State University
Computer Graphics

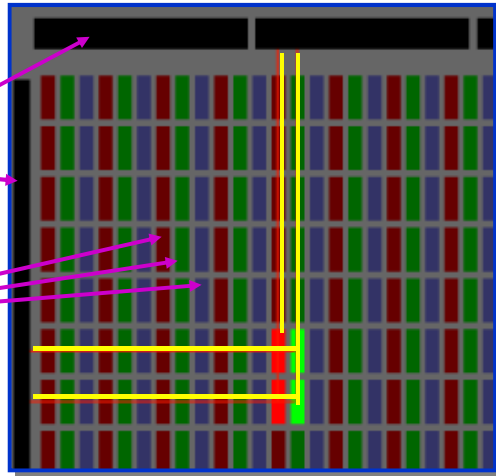
September 13, 2011

Brown Cunningham
Associates

Displaying Color on a Computer Graphics LCD Monitor



- Grid of electrodes
- Color filters



Source: <http://electronics.howstuffworks.com>



Oregon State University
Computer Graphics

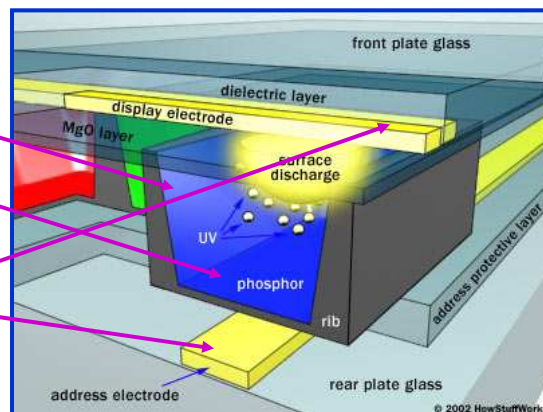
Brown Cunningham
Associates

September 13, 2011

Displaying Color on a Plasma Monitor



- Gas cell
- Phosphor
- Grid of electrodes



<http://electronics.howstuffworks.com>



Oregon State University
Computer Graphics

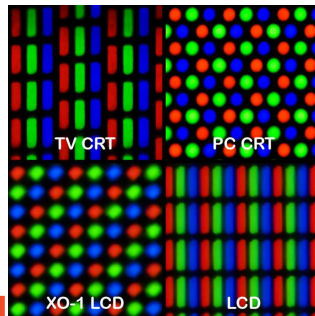
Brown Cunningham
Associates

September 13, 2011

Display Resolution



- **Pixel** resolutions (1024x768 - 1920x1152 are common)
- Screen size (13", 16", 19", 21" are common)
- Human acuity: 1 arc-minute is achieved by viewing a 19" monitor with 1280x1024 resolution from a distance of ~40 inches



http://en.wikipedia.org/wiki/File:Pixel_geometry_01_Pengo.jpg

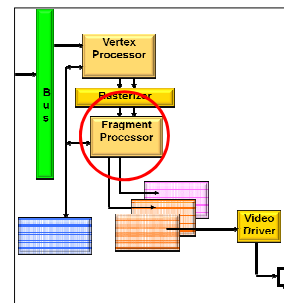
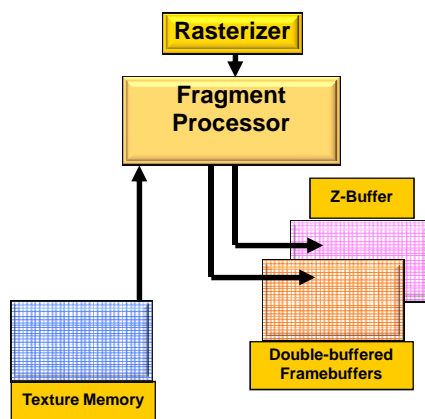
OSU

Oregon State University
Computer Graphics

Brown Cunningham
Associates

September 13, 2011

The Fragment Processor



OSU

Oregon State University
Computer Graphics

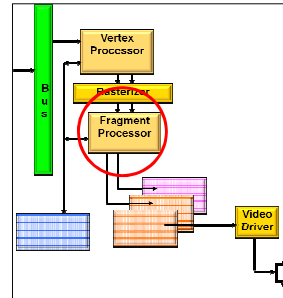
Brown Cunningham
Associates

September 13, 2011

The Fragment Processor



- Takes in all information that describes this pixel
- Produces the RGBA for that pixel's location in the framebuffer

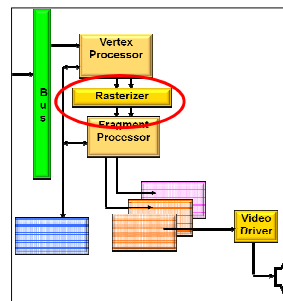
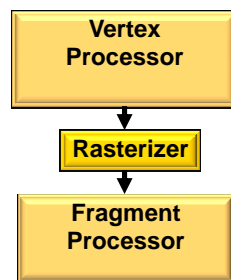


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

The Rasterizer



Oregon State University
Computer Graphics

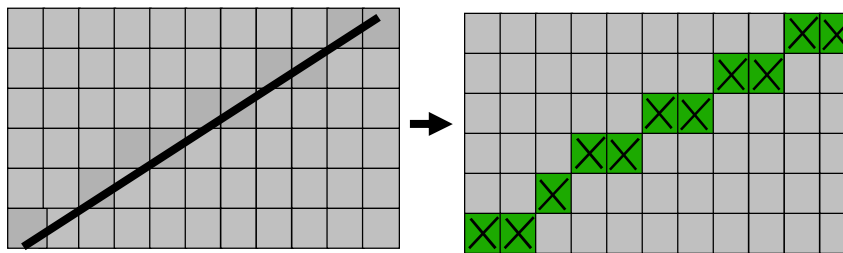
September 13, 2011

Brown Cunningham
Associates

Rasterization



- Turn screen space vertex coordinates into pixels that make up lines and polygons
- A great place for custom electronics



Oregon State University
Computer Graphics

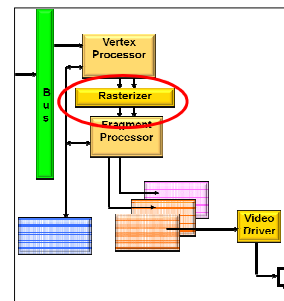
September 13, 2011

Brown Cunningham
Associates

Rasterizers Can Interpolate:



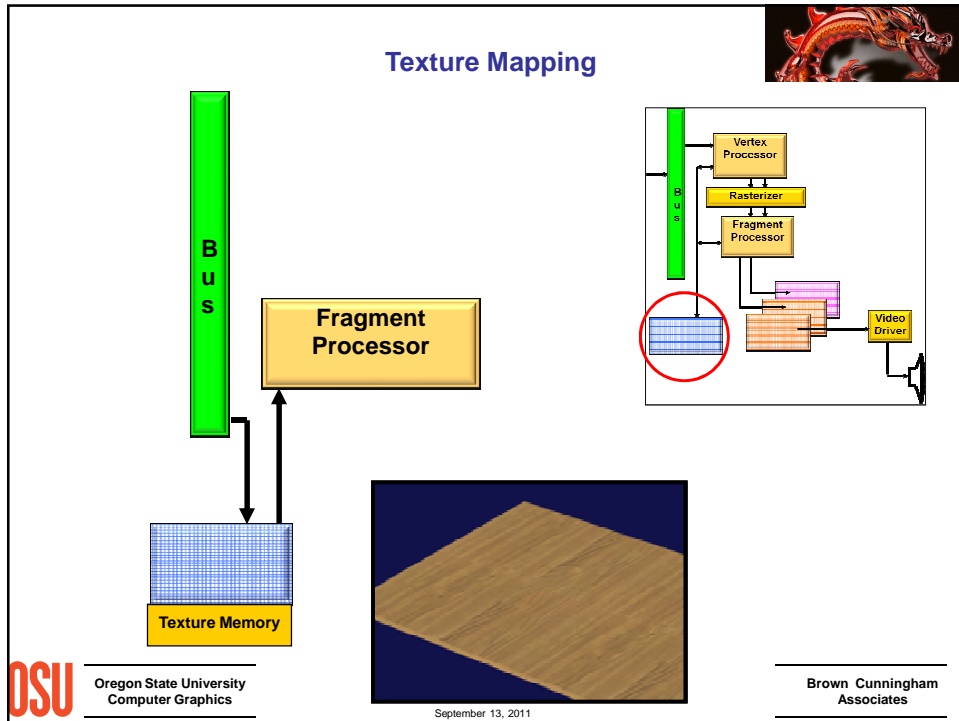
- X and Y
- Red-green-blue values
- Alpha values
- Z values
- Intensities
- Surface normals
- Texture coordinates
- Custom values given by the shaders



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates



Texture Mapping

- “Stretch” an image onto a piece of geometry
- Image can be generated by a program or scanned in
- Useful for realistic scene generation

The diagram shows a 3D terrain model with a texture map applied to it, demonstrating the result of texture mapping. The texture map is a grayscale image of a landscape, which is stretched onto the 3D geometry of the terrain. The diagram also includes a system architecture diagram on the right, similar to the one in the first slide, showing the flow from the "Vertex Processor" through the "Rasterizer" and "Fragment Processor" to the "Video Driver". A red circle highlights the "Texture Memory" in the system diagram.

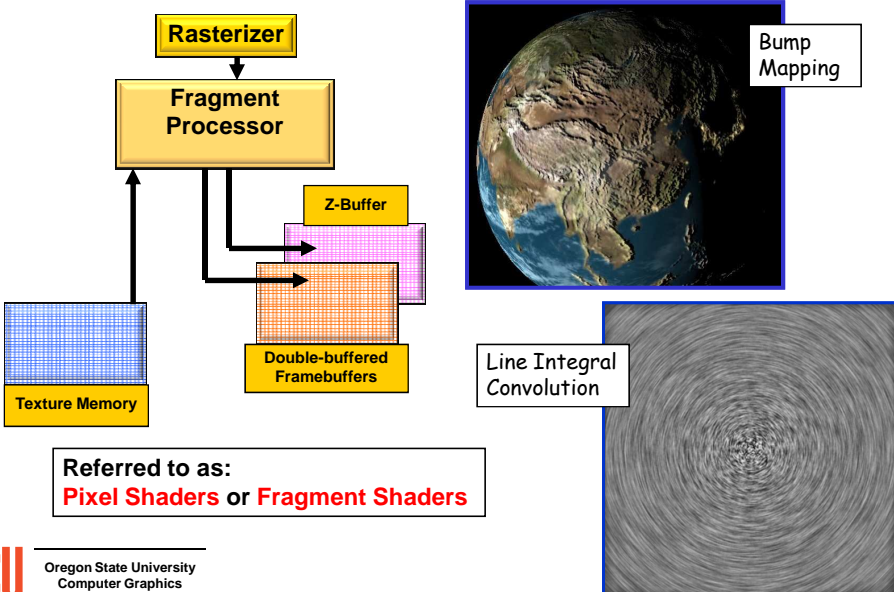
OSU Oregon State University
Computer Graphics

<http://2ols.com>


Brown Cunningham
Associates

September 13, 2011

Something Cool: Write-Your-Own Fragment-Processor Code



Referred to as:
Pixel Shaders or **Fragment Shaders**

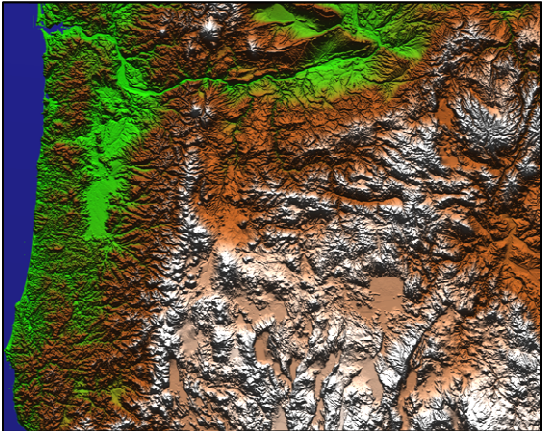



Oregon State University
Computer Graphics

September 13, 2011

Procedural Texture Mapping

Create a texture from an equation. In this case, the equation takes a grid of heights and produces surface normals for lighting



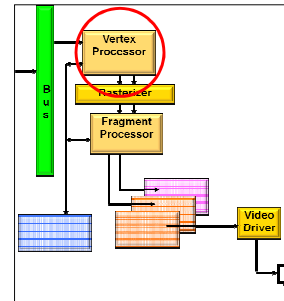
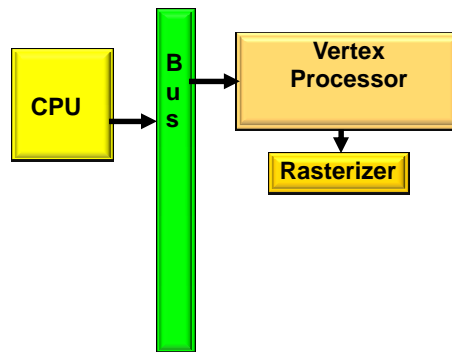


Oregon State University
Computer Graphics

Brown Cunningham
Associates

September 13, 2011

The Vertex Processor



Oregon State University
Computer Graphics

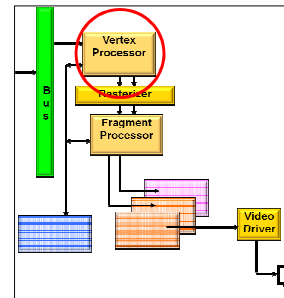
September 13, 2011

Brown Cunningham
Associates

Vertex Processor



- Coordinates enter in model units
- Coordinates leave in screen (pixel) units
- Another great place for custom electronics



Oregon State University
Computer Graphics

September 13, 2011

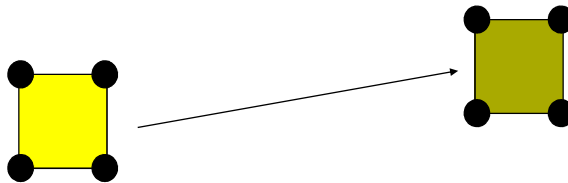
Brown Cunningham
Associates

Vertex Processor: Transformations

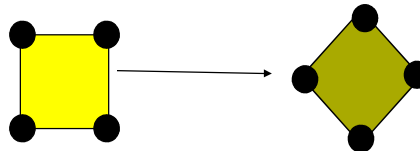


- Used to correctly place objects in the scene

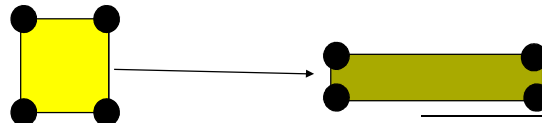
- Translation



- Rotation



- Scaling



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Vertex Processor: Windowing and Clipping



- Declare which portion of the 3D universe you are interested in viewing
- This is called the *view volume*
- Clip away everything that is outside the viewing volume



Oregon State University
Computer Graphics

September 13, 2011

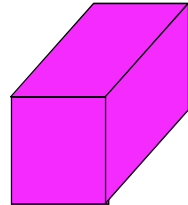
Brown Cunningham
Associates

Vertex Processor: Projection



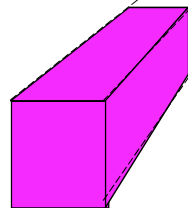
- Turn 3D coordinates into 2D

- *Parallel projection*



Parallel lines remain parallel

- *Perspective projection*



"vanishing point"

Some parallel lines appear to converge

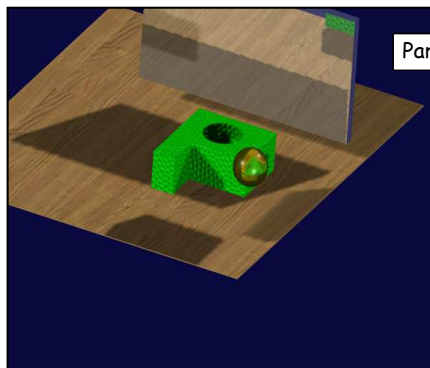


Oregon State University
Computer Graphics

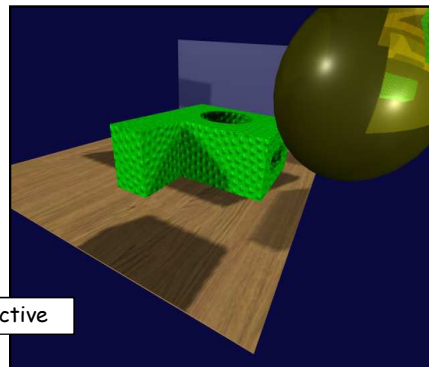
September 13, 2011

Brown Cunningham
Associates

Vertex Processor: Projection



Parallel



Perspective




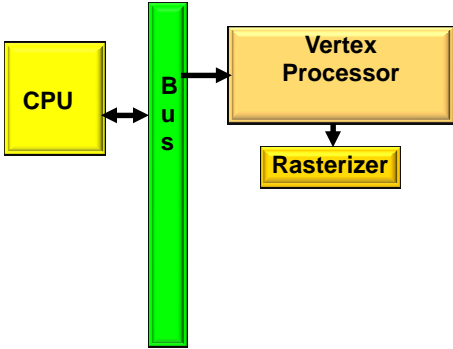
Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

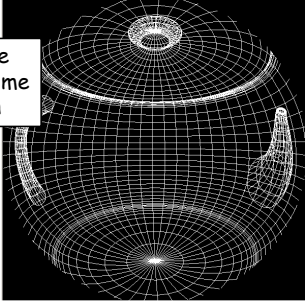
Something Cool: Write-Your-Own Vertex Code



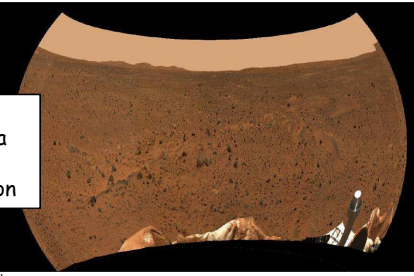



**Referred to as:
Vertex Shaders**

Wireframe
Teapot Dome
Projection



Mars
Panorama
Dome
Projection




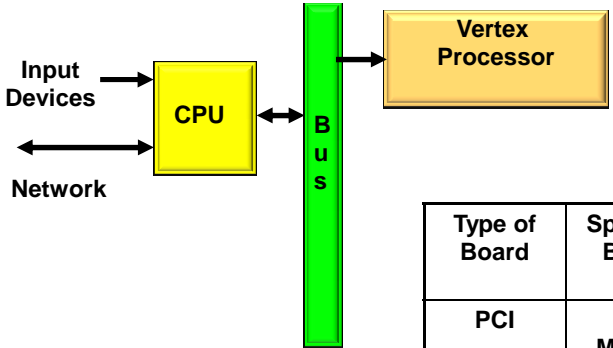


Oregon State University
Computer Graphics


September 13, 2011

The CPU and Bus





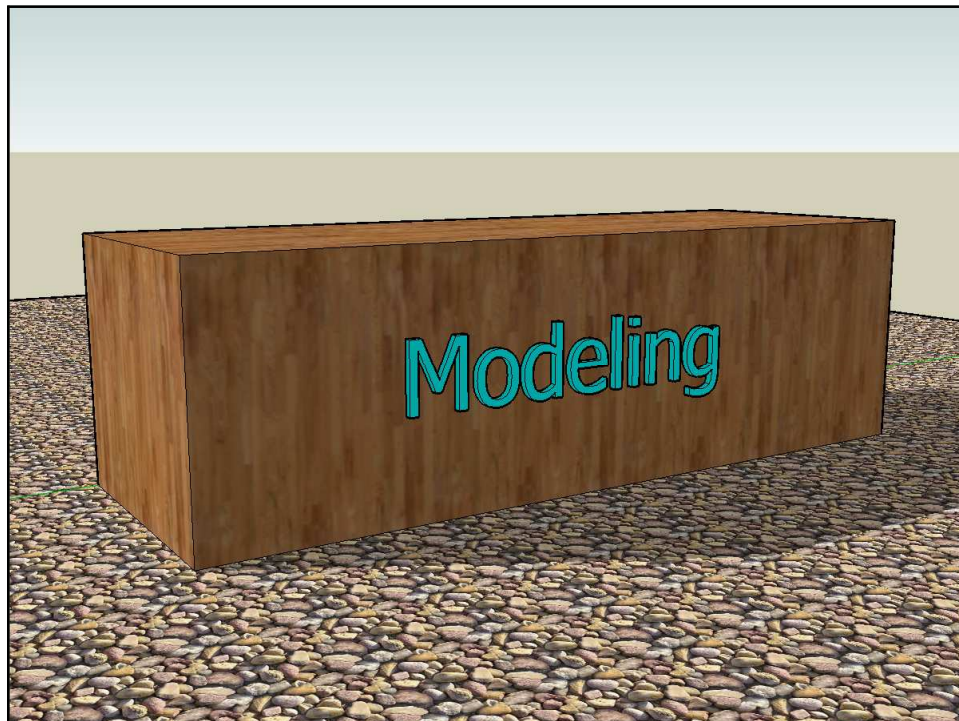
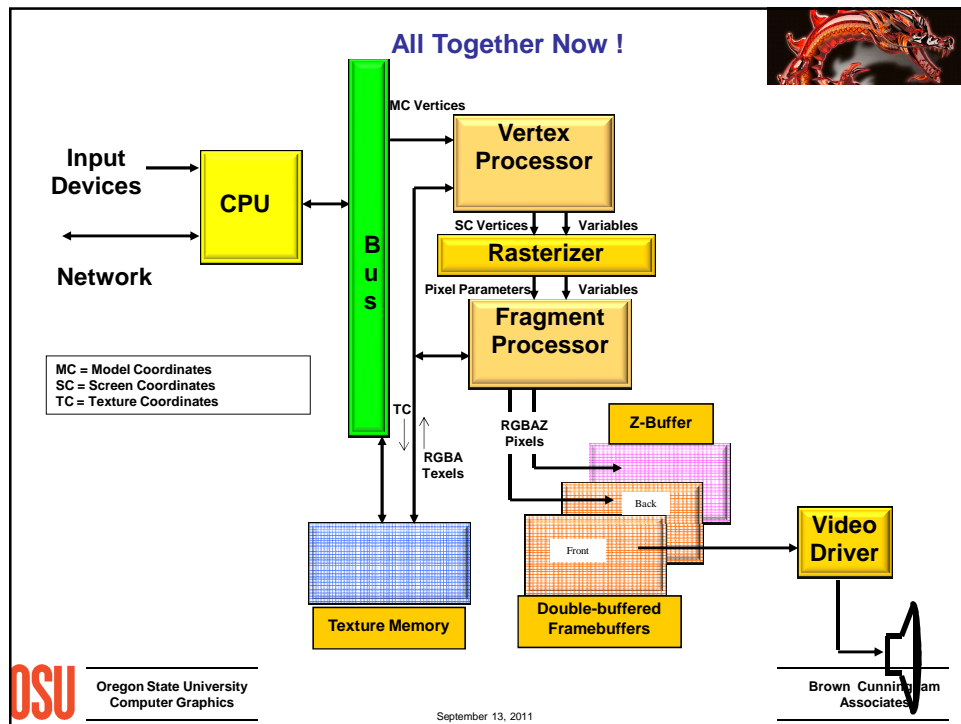
Type of Board	Speed to Board	Speed from Board
PCI	132 Mb/sec	132 Mb/sec
AGP 8X	2 Gb/sec	264 Mb/sec
PCI Express	4 Gb/sec	4 Gb/sec



Oregon State University
Computer Graphics

Brown Cunningham
Associates

September 13, 2011



What is a Model?



A is a model of B if A can be used to ask questions about B.

In computer graphics applications, what do we want to ask about B?

- What does B look like?
- How do I want to interact with (shape) B?
- Does B need to be a legal solid?
- How does B interact with its environment?
- What is B's surface area and volume?

These questions, and answers, control what type of geometric modeling you need to do



Oregon State University
Computer Graphics

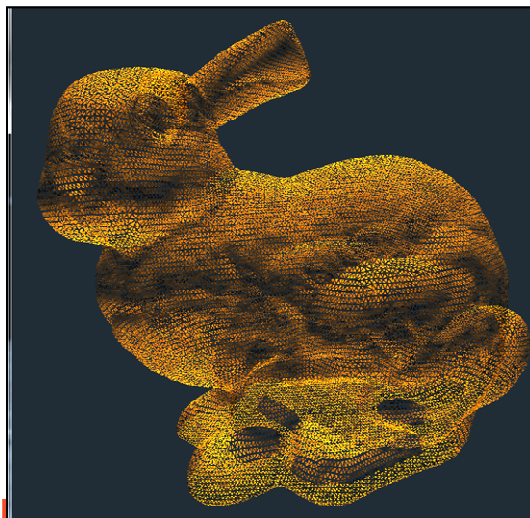
September 13, 2011

Brown Cunningham
Associates

Explicitly Listing Geometry and Topology



Models can consist of thousands of vertices and faces – we need some way to list them efficiently



<http://graphics.stanford.edu/data/3Dscanrep>

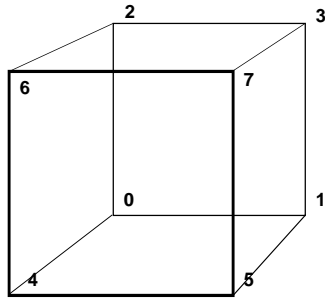


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Explicitly Listing Geometry and Topology



```
static GLfloat CubeVertices[ ][3] =
{
    { -1., -1., -1. },
    { 1., -1., -1. },
    { -1., 1., -1. },
    { 1., 1., -1. },
    { -1., -1., 1. },
    { 1., -1., 1. },
    { -1., 1., 1. },
    { 1., 1., 1. }
};
```

```
static GLfloat CubeColors[ ][3] =
{
    { 0., 0., 0. },
    { 1., 0., 0. },
    { 0., 1., 0. },
    { 1., 1., 0. },
    { 0., 0., 1. },
    { 1., 0., 1. },
    { 0., 1., 1. },
    { 1., 1., 1. },
};
```

```
static GLuint CubeIndices[ ][4] =
{
    { 0, 2, 3, 1 },
    { 4, 5, 7, 6 },
    { 1, 3, 7, 5 },
    { 0, 4, 6, 2 },
    { 2, 6, 7, 3 },
    { 0, 1, 5, 4 }
};
```

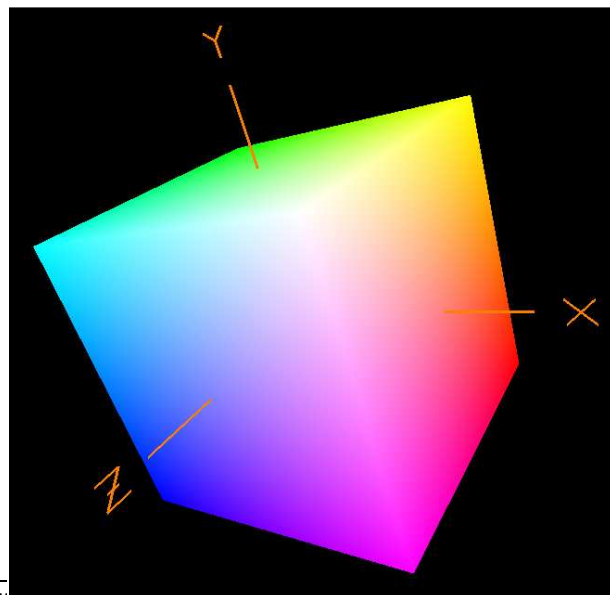


Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Cube Example

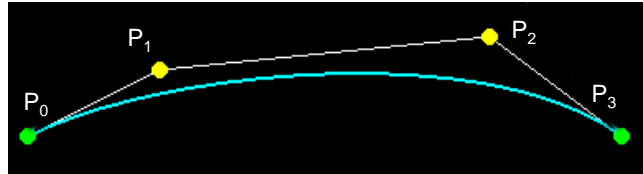


Oregon State U
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Curve Sculpting – Bezier Curve Sculpting



$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

$$0 \leq t \leq 1.$$

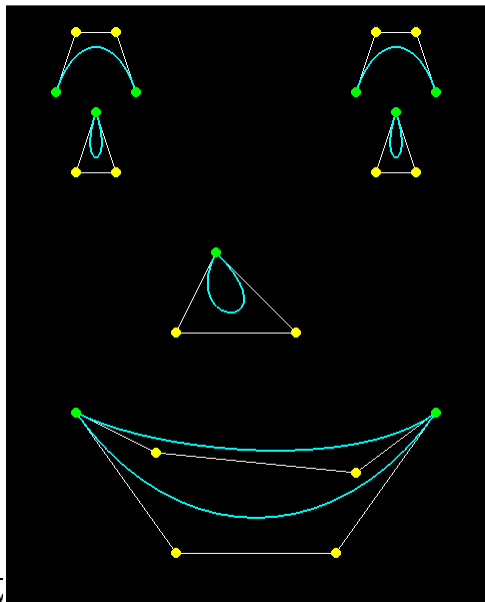


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Curve Sculpting – Bezier Curve Sculpting Example

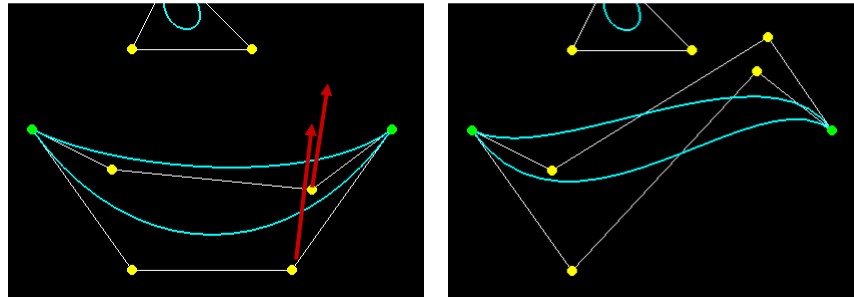


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Curve Sculpting – Bezier Curve Sculpting Example

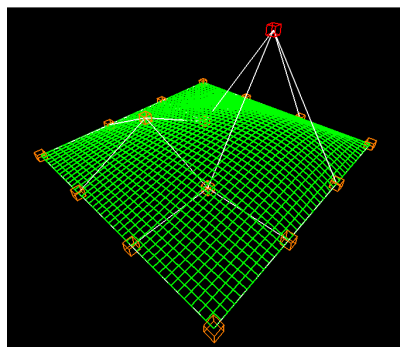


Oregon State University
Computer Graphics

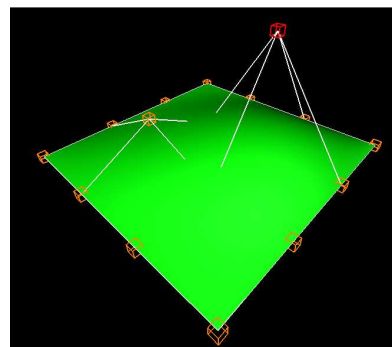
September 13, 2011

Brown Cunningham
Associates

Surface Sculpting



Wireframe



Surface

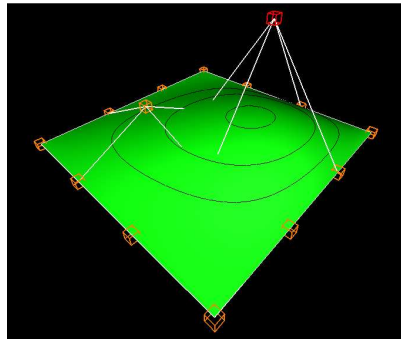


Oregon State University
Computer Graphics

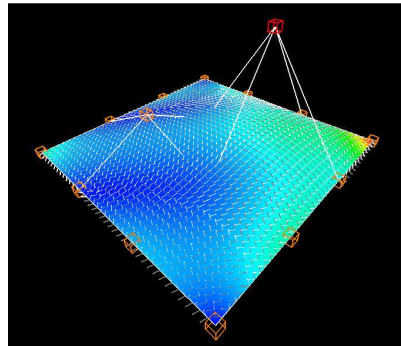
September 13, 2011

Brown Cunningham
Associates

Surface equations can also be used for Analysis



With Contour Lines



Showing Curvature

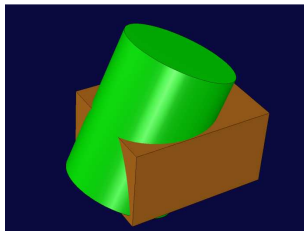


Oregon State University
Computer Graphics

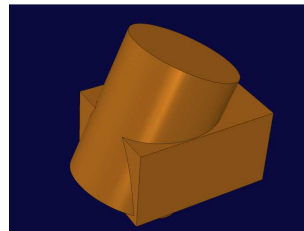
September 13, 2011

Brown Cunningham
Associates

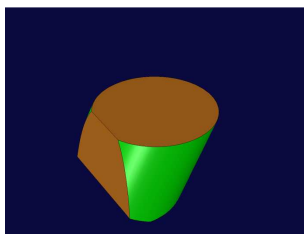
Solid Modeling Using Boolean Operators



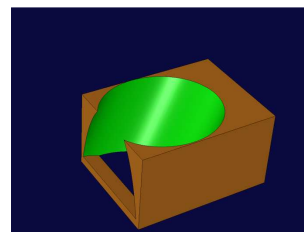
Two Overlapping Solids



Union



Intersection



Difference



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates



Rendering



Rendering is the process of creating an image of a geometric model. Again, there are questions you need to ask:

- How realistic do I want this image to be?
- How much compute time do I have to create this image?
- Do I need to take into account lighting?
- Does the illumination need to be global or will local do?
- Do I need to take into account shadows?
- Do I need to take into account reflection and refraction?

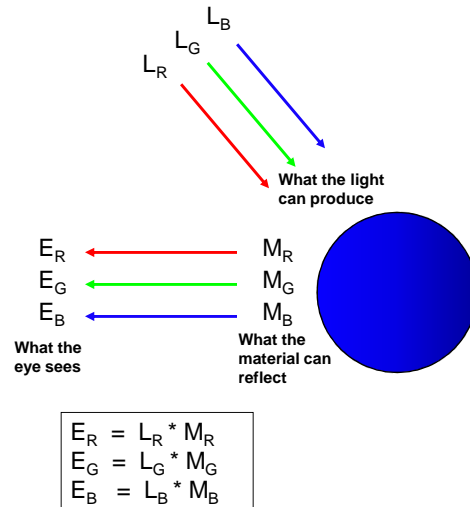


Oregon State University
Computer Graphics

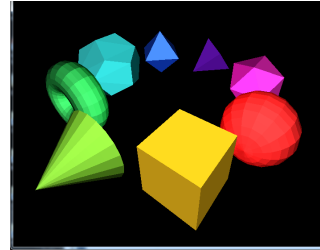
September 13, 2011

Brown Cunningham
Associates

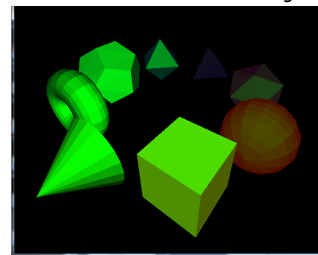
Fundamentals of Computer Graphics Lighting



White Light



Green Light

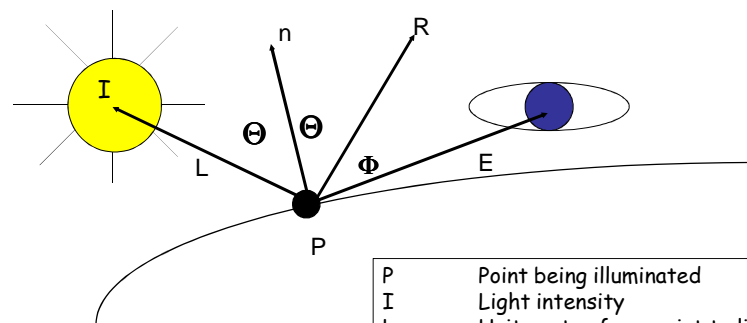


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

The Computer Graphics Lighting Environment



P Point being illuminated
I Light intensity
L Unit vector from point to light
n Unit vector surface normal
R Perfect reflection unit vector
E Unit vector to eye position



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Three Elements of Computer Graphics Lighting



1. Ambient = a constant Accounts for light bouncing "everywhere"
2. Diffuse = $I \cdot \cos\Theta$ Accounts for the angle between the incoming light and the surface normal
3. Specular = $I \cdot \cos^S\phi$ Accounts for the angle between the "perfect reflector" and the eye; also the exponent, S , accounts for surface shininess

Note that $\cos\Theta$ is just the dot product between unit vectors L and n

Note that $\cos\phi$ is just the dot product between unit vectors R and E



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates



+



+



=



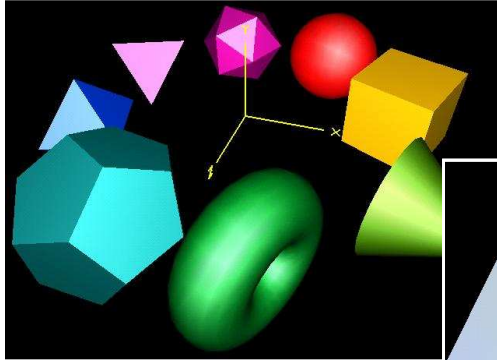
Three Elements of Computer Graphics Lighting



Oregon State University
Computer Graphics

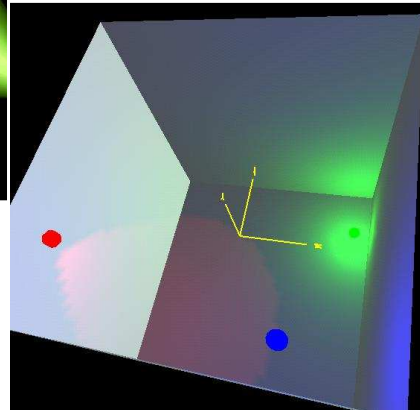
September 13, 2011

Lighting Examples



Omnidirectional Point Light

Spot Lights



Brown Cunningham Associates



Oregon State University
Computer Graphics

September 13, 2011

Two Types of Rendering



1. Starts at the object
2. Starts at the eye



Oregon State University
Computer Graphics

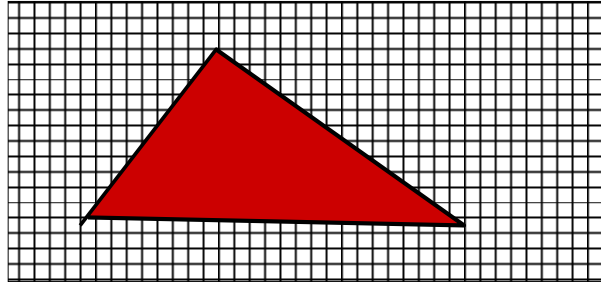
September 13, 2011

Brown Cunningham
Associates

Starts at the Object



This is the typical kind of rendering you get on a graphics card.
Start with the geometry and project it onto the pixels.



Oregon State University
Computer Graphics

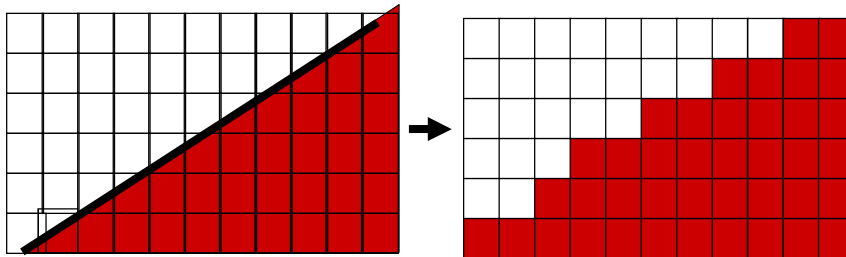
September 13, 2011

Brown Cunningham
Associates

Rasterization



- Turn screen space vertex coordinates into pixels that make up lines and polygons
- A great place for custom electronics



Oregon State University
Computer Graphics

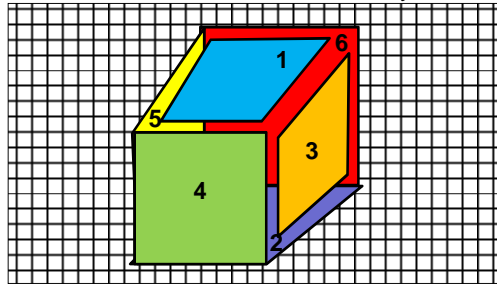
September 13, 2011

Brown Cunningham
Associates

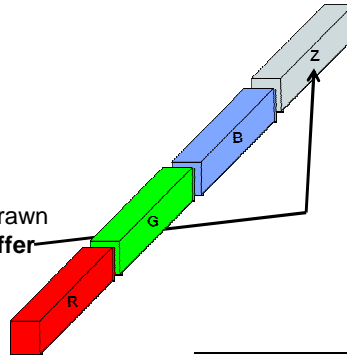
How do things in front *look* like they are really in front?



Your application might draw the polygons in 1-2-3-4-5-6 order, but 1, 3, and 4 still need to look like they were drawn last:



Either the polygons need to be re-arranged to be drawn in a back-to-front order, or we need to have a **Z-buffer**



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Another From-the-Object Method -- Radiosity

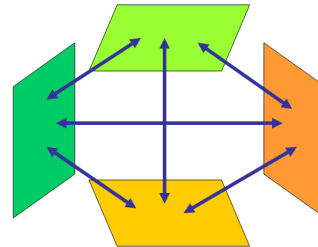


Based on the idea that all surfaces gather light intensity from all other surfaces

The fundamental radiosity equation is an energy balance that says:

"The light energy leaving surface i equals the amount of light energy generated by surface i plus surface i 's reflectivity times the amount of light energy arriving from all other surfaces"

$$B_i A_i = E_i A_i + \rho_i \sum_j B_j A_j F_{j \rightarrow i}$$



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

The Radiosity Equation



$$B_i A_i = E_i A_i + \rho_i \sum_j B_j A_j F_{j \rightarrow i}$$

B_i is the light energy intensity shining from surface element i

A_i is the area of surface element i

E_i is the internally-generated light energy intensity for surface element i

ρ_i is surface element i 's reflectivity

$F_{j \rightarrow i}$ is referred to as the Form Factor, or Shape Factor, and describes what percent of the energy leaving surface element j that arrives at surface element i

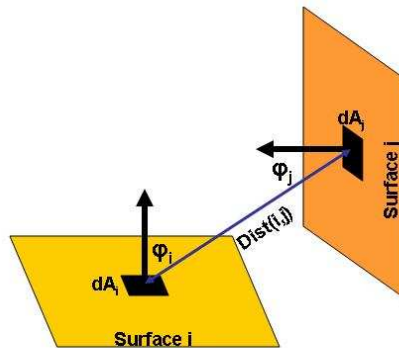


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

The Radiosity Shape Factor



$$F_{j \rightarrow i} = \int \int_{A_i A_j} \text{visibility}(di, dj) \frac{\cos \Theta_i \cos \Theta_j}{\pi \text{Dist}(di, dj)^2} dA_j dA_i$$



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

The Radiosity Matrix Equation



Expand $B_i A_i = E_i A_i + \rho_i \sum_j B_j A_j F_{j \rightarrow i}$

For each surface element, and re-arrange to solve for the surface intensities, the B 's:

$$\begin{bmatrix} 1 - \rho_1 F_{1 \rightarrow 1} & -\rho_1 F_{1 \rightarrow 2} & \cdots & -\rho_1 F_{1 \rightarrow N} \\ -\rho_2 F_{2 \rightarrow 1} & 1 - \rho_2 F_{2 \rightarrow 2} & \cdots & -\rho_2 F_{2 \rightarrow N} \\ \cdots & \cdots & \cdots & \cdots \\ -\rho_N F_{N \rightarrow 1} & -\rho_N F_{N \rightarrow 2} & \cdots & 1 - \rho_N F_{N \rightarrow N} \end{bmatrix} \begin{Bmatrix} B_1 \\ B_2 \\ \cdots \\ B_N \end{Bmatrix} = \begin{Bmatrix} E_1 \\ E_2 \\ \cdots \\ E_N \end{Bmatrix}$$

This is a lot of equations!



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Radiosity Examples



AR Toolkit



Autodesk



Oregon State University
Computer Graphics

September 13, 2011

Radiosity Examples



Cornell University



Brown Cunningham
Cornell University



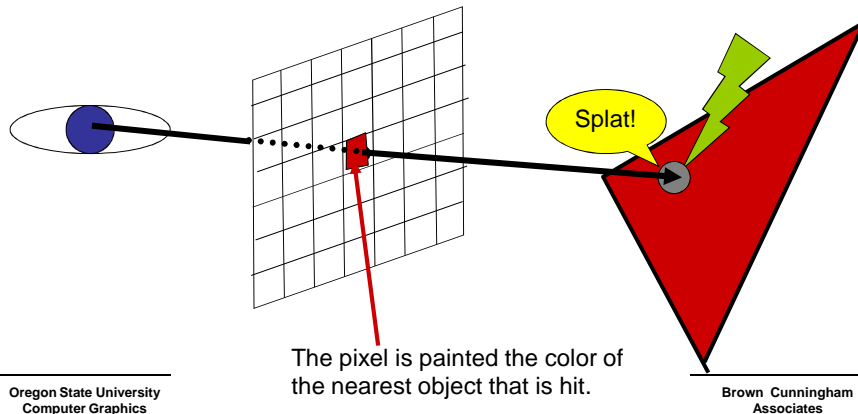
Oregon State University
Computer Graphics

September 13, 2011

Starts at the Eye



The most common approach in this category is ray-tracing:



Oregon State University
Computer Graphics

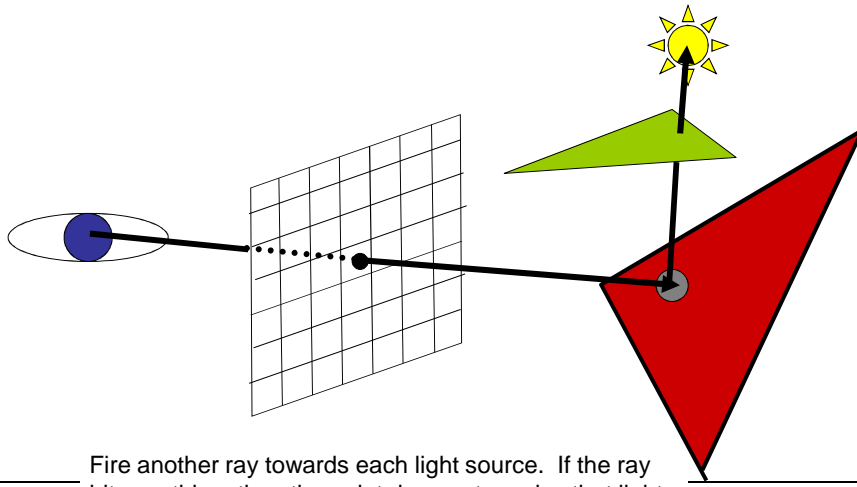
September 13, 2011

Brown Cunningham
Associates

Starts at the Eye



It's also easy to see if this point lies in a shadow:



Oregon State
Computer Graphics

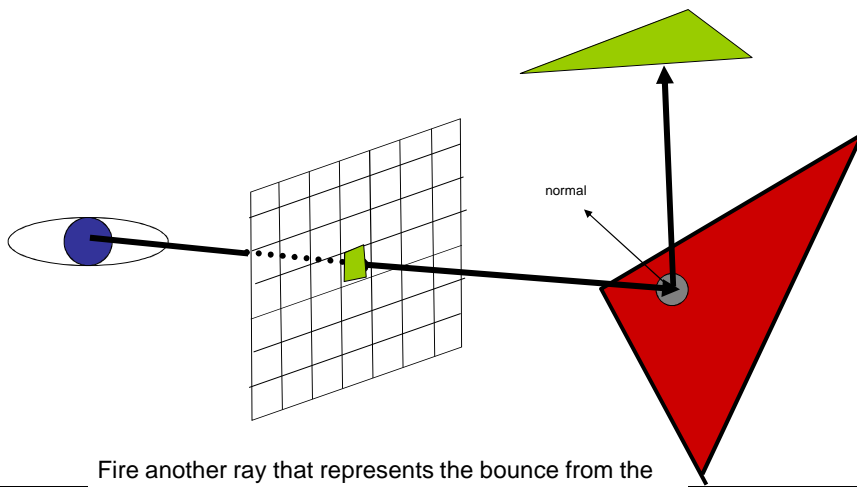
September 13, 2011

Brown Cunningham
Associates

Starts at the Eye



It's also easy to handle reflection



Oregon State
Computer Graphics

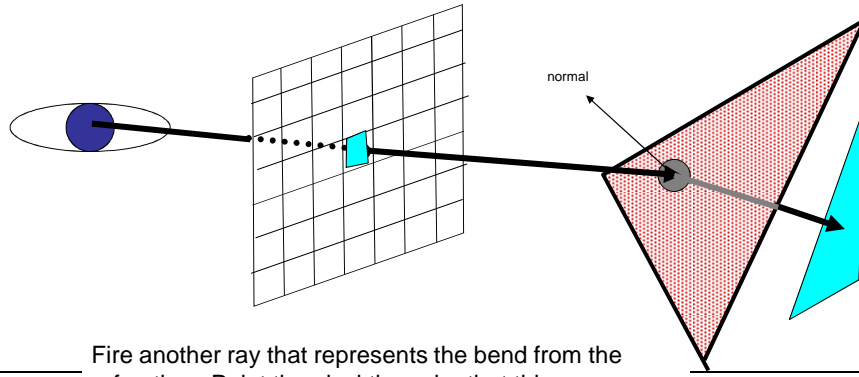
September 13, 2011

Brown Cunningham
Associates

Starts at the Eye



It's also easy to handle refraction

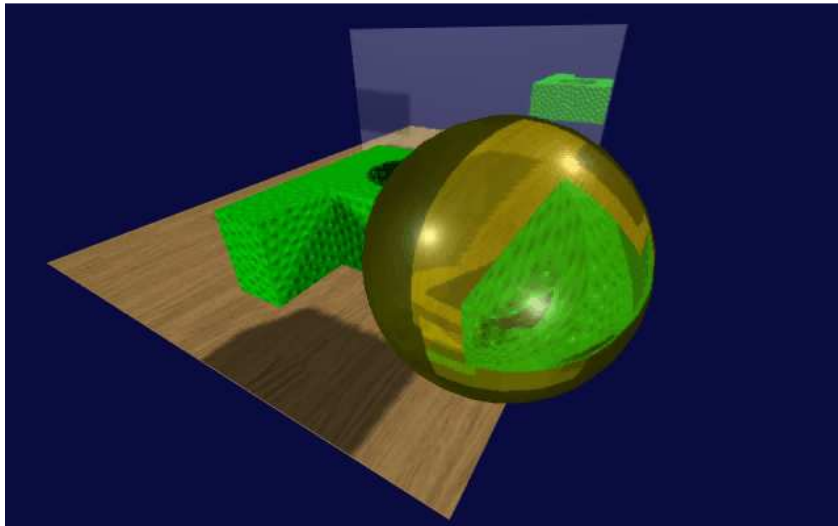


Oregon State
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Ray Tracing Examples



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Ray Tracing Examples



Quake 4 Ray-Tracing Project



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Ray Tracing Examples



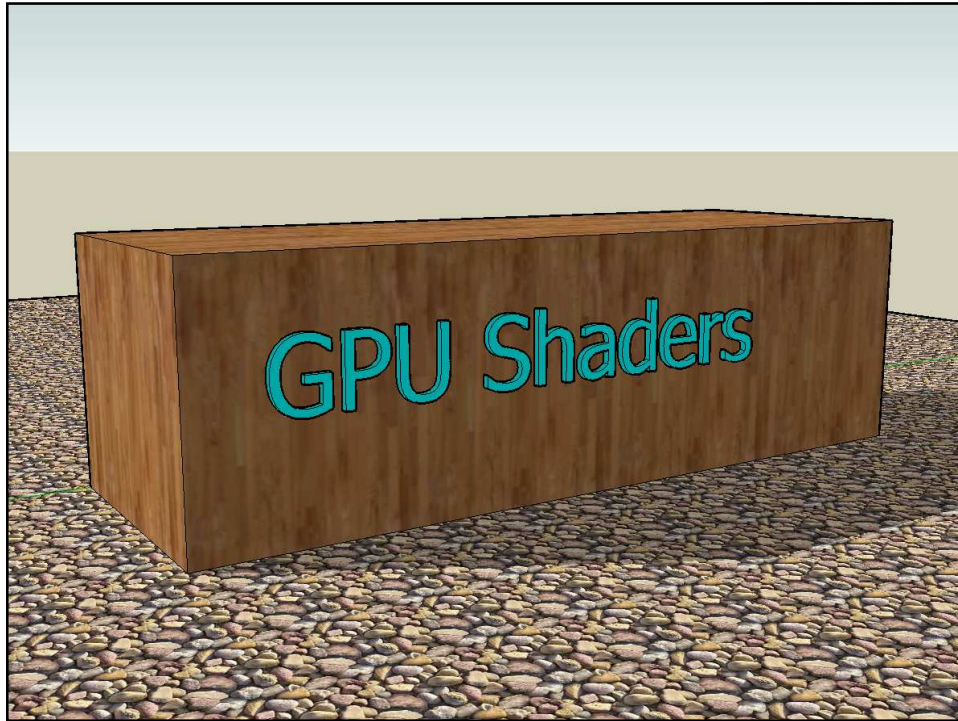
IBM's Cell Interactive Ray-tracer



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates



GPU Shader Programming



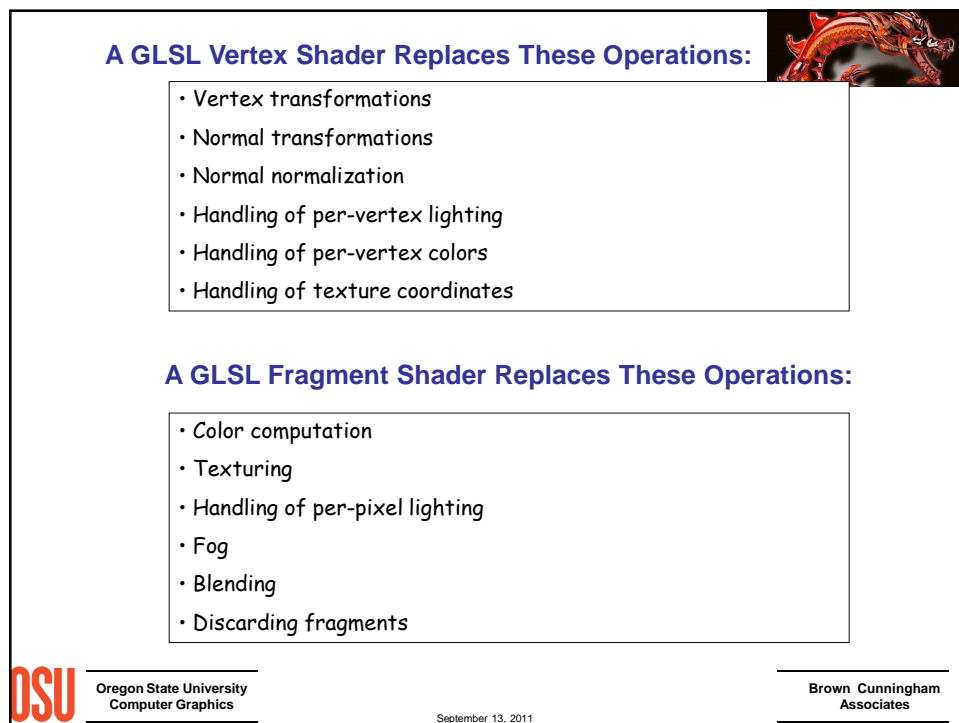
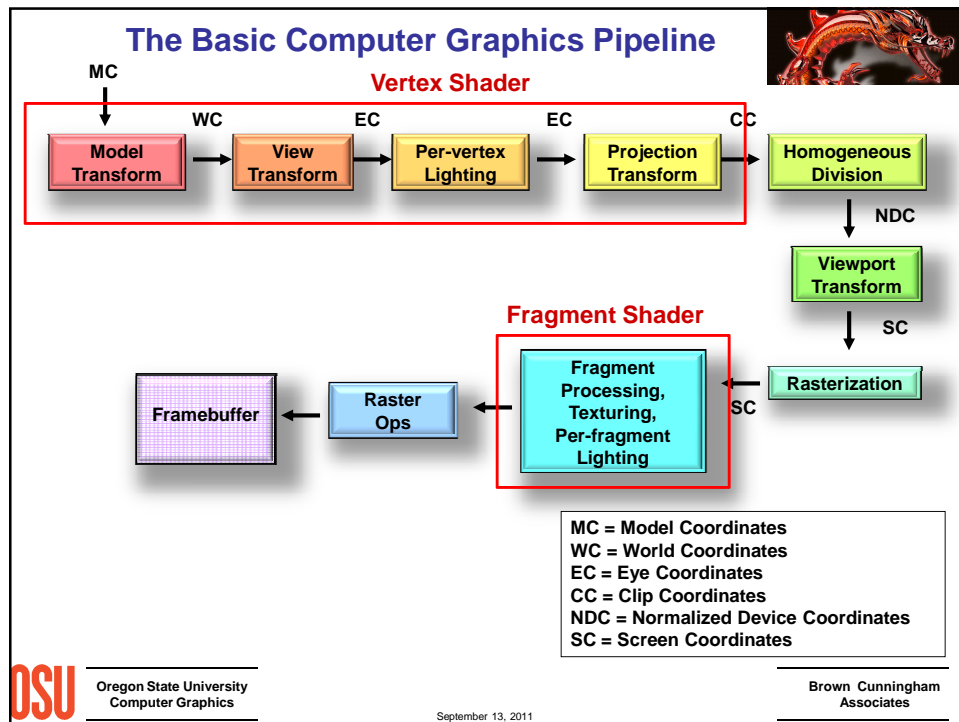
- Allows programmers to load their own code into parts of the hardware graphics pipeline
- Gives a unique combination of control and speed
- This is a hot, new area in computer graphics
- These notes will focus on *what* can be done this way, not on *how* to do it (that would take lots more time)
- If you want to know more, there's another course on just this topic!



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates



A GLSL Tessellation Shader:



- Breaks geometry into smaller pieces based on adjacent points, size, curvature, etc.

A GLSL Geometry Shader:

- Breaks geometry into smaller pieces based on more limited information
- Changes the geometry's topology type
- Changes the object's coordinates

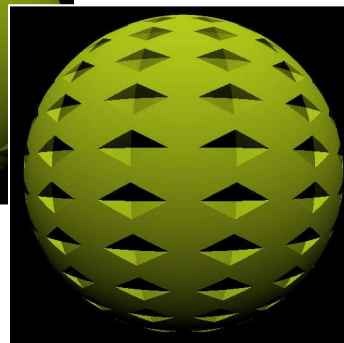
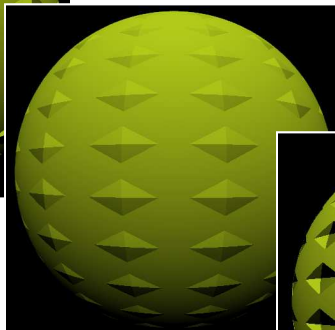
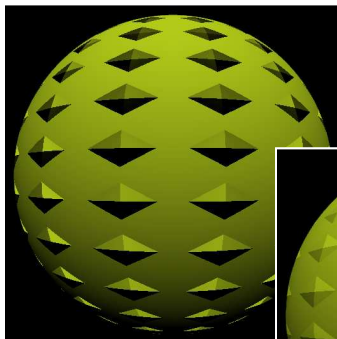


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Bump Mapping with Shaders

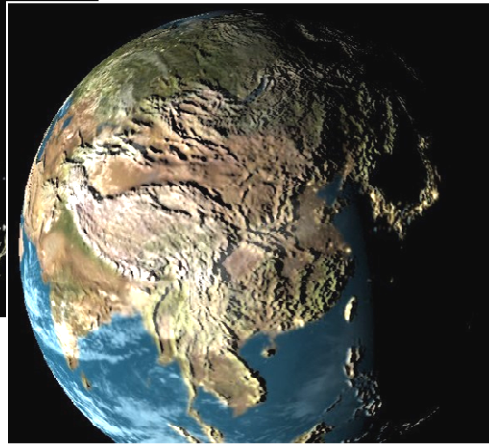
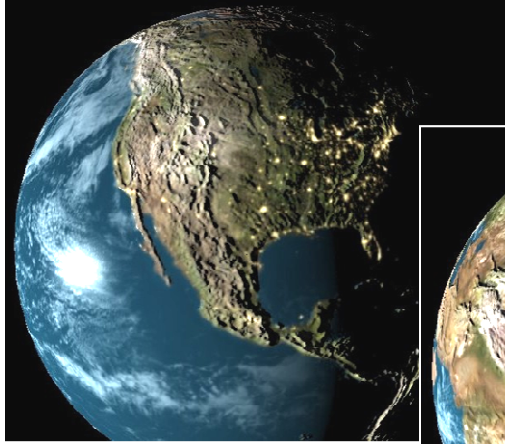


Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Bump Mapping with Shaders



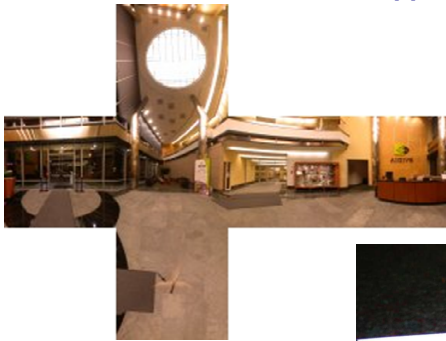
Visualization by Nick Gebbie



Oregon State University
Computer Graphics

September 13, 2011

Cube Mapping with Shaders



Cube Map of NVIDIA's Lobby



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Cube Mapping with Shaders



Oregon State University
Computer Graphics

September 13, 2011

Brown Cunningham
Associates

Cube Mapping with Shaders

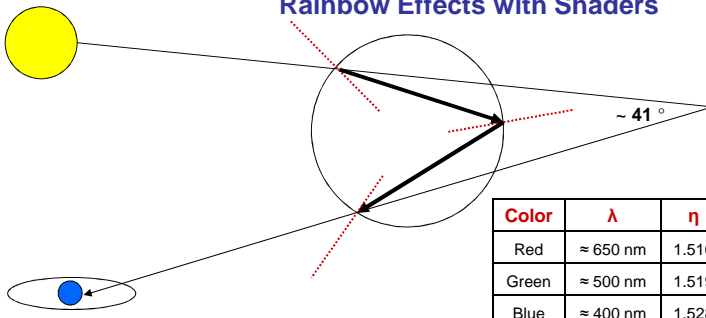


Oregon State University
Computer Graphics

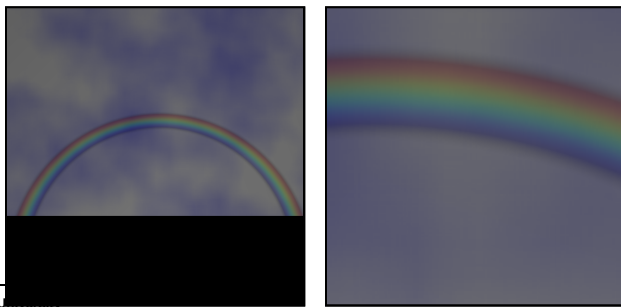
September 13, 2011

Brown Cunningham
Associates

Rainbow Effects with Shaders



Color	λ	η	Θ	$\cos\Theta$	$\Theta\Theta$
Red	$\approx 650 \text{ nm}$	1.510	42°	0.743	50.0°
Green	$\approx 500 \text{ nm}$	1.519	41°	0.755	51.5°
Blue	$\approx 400 \text{ nm}$	1.528	40°	0.766	53.0°



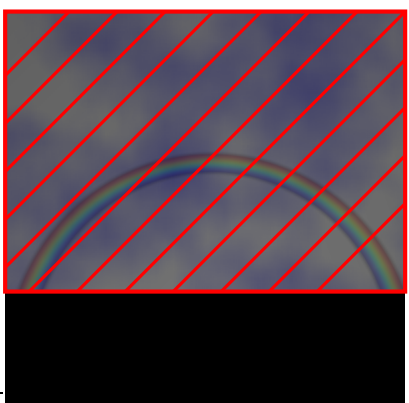
OSU Oregon State University Computer Graphics

Brown Cunningham Associates

September 13, 2011

Rainbow Strategy

1. Draw one big quadrilateral across the scene
2. Anywhere that $.7400 \leq \cos(\Theta) \leq .7700$, paint the correct color
3. If not, discard that fragment



OSU Oregon State University Computer Graphics

Brown Cunningham Associates

September 13, 2011

